DOCUMENT RESUME

ED 287 704 SE 048 659

AUTHOR	Murray, Tom; Woolf, Beverly
TITLE	Knowledge Representation in a Physics Tutor. COINS Technical Report 86-37.
INSTITUTION	Massachusetts Univ., Amherst. Dept. of Computer and Information Science.
PUB DATE	86
NOTE	20p.
PUB TYPE	Reports - Descriptive (141)
EDRS PRICE	MF01/PC01 Plus Postage.
DESCRIPTORS	Cognitive Processes; Cognitive Structures; *College Science; Higher Education; *Physics; Problem Solving; Science Activities; Science Education; *Science Instruction; Secondary Education; *Secondary School Science; *Student Characteristics; *Tutors

ABSTRACT

This paper is based on the idea that designing a knowledge representation for an intelligent physics computer tutoring system depends, in part, on the target behavior anticipated from the student. In addition, the document distinguishes between qualitative and quantitative competence in physics. These competencies are illustrated through questions that students are expected to answer for two problems (a crane boom and a stone throw problem) and approaches are described that are consistent with each type of competence. An example is described which demonstrates that an approach that leads to competence in qualitative reasoning emphasizes pedagogic and conceptual knowledge, and an approach that leads to competence in quantitative reasoning is an expert system which emphasizes problem solving and factual knowledge. Two ways to categorize the knowledge of a physics tutor are presented. The first consists of facts, skills, and concepts, and the second distinguishes between expert and pedagogic knowledge. Several figures are provided. (TW)

KNOWLEDGE REPRESENTATION IN A PHYSICS TUTOR

Tom Murray
Beverly Woolf

COINS Technical Report 86-37

ABSTRACT

Designing a knowledge representation for an intelligent tutor depends in part, on the target behavior anticipated from the student and we distinguish between competence in qualitative physics and competence in quantitative. We illustrate this competence through questions we expect a student to be able to answer for two example problems, a crane boom and a stone throw problem and describe approaches consistent with each type of competence. For example, the approach that leads to competence in qualitative reasoning emphasizes pedagogic and conceptual knowledge and the approach that leads to competence in quantative reasoning, is an expert system which emphasizes problem solving and factual knowledge. In establishing a vocabulary for discussing knowledge representation issues, we suggest two (orthogonal) ways to categorize the knowledge of a physics tutor. The first consists of facts, skills, and concepts and the second distinguishes between expert and pedagogic knowledge.

KNOWLEDGE REPRESENTATION IN A PHYSICS TUTOR

Tom Murray
Beverly Woolf

Department of Computer and Information Science
University of Massachusetts
Amherst, Massachusetts, 01003
1 July 1986

1. Introduction

We are on the verge of developing substantially more powerful intelligent tutoring systems that can reason about a student's knowledge and custom-tailor their teaching strategy to his individual learning pattern. Such systems will be able to simulate "worlds" (e.g., the ocean, atmosphere, power plants, ecosystems, etc.) in a visually rich and informationally dense way that is not currently possible.

Obviously, we are not yet capable of building such systems; formidable barriers, both hardware and software, stand between us and full realization of the potential. However, many of these barriers are theoretical, rather than engineering; i.e., they depend on providing new abilities or new results to the computer. In this article we discuss one of the most salient of these barriers, representing the knowledge for a computer tutor. By this we mean teasing apart, and codifying, knowledge of the domain, discourse, tutoring, and of the student for use inside the computer.

The very fact that knowledge representation remains a barrier to successful development of intelligent tutors underscores the important difference between intelligent teaching systems and computer-aided instruction, CAI. Since intelligent tutoring systems reason about the student and the domain, they require encoded knowledge of the student, tutoring, and of discourse conventions to make reasonable decisions about their responses to the student.

The purpose of this article is to establish a vocabulary -- a common ground -- for the various participants in the development phase of a computer tutor. These participants include, but are not limited to, teachers, computer scientists, psychologists, and domain specialists. Each will need to work with the others to build intelligent tutoring systems: teachers will have to become familiar with the knowledge engineering elements of a system and the computer scientists with the the educator's expertise in academic domains, in pedagogy, and in curriculum design. In developing this common ground vocabulary, we are additionally addressing all the concerns of knowledge representation that impact on the development of tutoring systems.

The Exploring Systems Earth (ESE) has already begun to train high school science teachers in knowledge engineering issues. For collaborations such as there to be effective, the parties involved must work from a common vocabulary and structure that acts as the basis for communication and the evolution of ideas. The Umass ITS group is working toward such a representational scheme for intelligent tutors.

There are several criteria for developing a general knowledge representation scheme for tutoring. Such a representation must 1) be general enough to be used for tutoring many domains and many types of knowledge (facts, skills, etc.), 2) be powerful enough to support (or upgrade to) the sophisticated inferencing needed for expert problem solving and student behavior diagnosis, and 3) clear and unambiguous enough to make the task of transferring a teacher's knowledge to the computer efficient and straightforward.

This paper describes our current work on developing such a knowledge representation (KR) scheme. The paper focuses on the domain of physics, but the intent is to design a general KR scheme that could be used in any science or technical domain. The KR scheme is intended to represent only domain specific knowledge (physics in this case); we do not address general knowledge about tutoring rules or discourse conventions (see Woolf 1984 for a discussion of these). We do, however, include what we call pedagogic components of domain knowledge. Pedagogic knowledge is the knowledge that a tutor needs to teach expert knowledge. Expert knowledge is knowledge needed to solve a problem in the domain. Examples of pedagogic knowledge are a hierarchy of salient topics, pointers to useful examples, and information about how the domain knowledge is organized pedagogicly.

We address two issues in the design of a knowledge representation (KR) scheme:

* The KR design depends, in part, on the target behavior anticipated from the student and we distinguish between competence in *qualitative* physics and competence in *quantitative*. We illustrate this competence through questions we expect a student to be able to answer for two example problems, a crane boom and a stone throw problem and describe approaches consistent with each type of competence. For example, the approach that leads to competence in qualitative reasoning emphasizes pedagogic and conceptual knowledge and the approach that leads to competence in quantitative reasoning, is an expert system which emphasizes problem solving and factual knowledge.

* In establishing a vocabulary for discussing KR issues, we suggest two (orthogonal) ways to categorize the knowledge of a physics tutor. The first consists of facts, skills, and concepts and the second distinguishes between expert and pedagogic knowledge.

2. Target Behaviors for Quantitative vs. Qualitative Understanding

The structure and content of a computer tutor, and thus its knowledge representation (KR), strongly depends on the types of target behaviors one wishes to see in students. In this section we look at performance on quantitative vs. qualitative physics questions that we expect our students to have and share our speculations about what a KR would look like for tutors focusing on either behavior. We will be as clear as possible about the intended use of the system.

Both quantitative and qualitative questions (and both types of reasoning) are needed in learning physics. Qualitative questions encourage the student to think about a physics situation in realistic, non-formular-centered ways; such questions are useful for enabling the student to diagnose and construct her conceptual knowledge. Quantitative questions are especially useful for diagnosing and improving problem solving abilities.

Most questions in standard physics homework exercises and exams are of the quantitative type. Often students who appear competent on such questions have difficulties solving non-standard problems (of both qualitative and quantitative types). These difficulties can be manifestations of poor problem solving abilities and/or inadequate grasp of the basic concepts.

Qualitative target behaviors include a student's ability to answer questions about:

- the existence of objects, properties, and variables (ex: Does a force exits here?);
- the relative magnitudes of variables (ex: Which side has the larger force?);
- the directions of vectors (up, down, etc;) and changes (greater or smaller);
- features of items, compare and contrast;
- causality, functionality and importance relationships;
- hierarchical relationships such as part-whole, classification, and set membership;

We are currently more concerned with the first three of these in teaching physics knowledge.

Quantitative target behaviors include a student's ability to:

- determine numerical values for variables given a sufficient set of facts;
- demonstrate domain-specific problem solving skills, procedures, and heuristics;

We can best illustrate our notions about the two types of reasoning through examples and questions used for each. We use two physical situations, throwing a stone vertically and a simple crane-boom problem (see Figures 1 & 2). Corresponding to each behavior are specific questions that a tutor might ask to test that a student has mastered the particular approach. Qualitative and quantitative questions for each example are listed in Figure 3 and for the crane boom in Figure 4. Note that some of the qualitative question are intended to check misconceptions

One can appreciate that the structure and content of the knowledge representation needed for each type of reasoning and teaching should be quite different. In the third section we present our design suggestions for a qualitative and quatitative tutor and continue to describe each system as a separate entity. This is for illustration purposes only – an actual physics tutor should incorporate the reasoning and representation of both kinds of tutors. In the next section we present a taxonomy of the types of knowledge used in

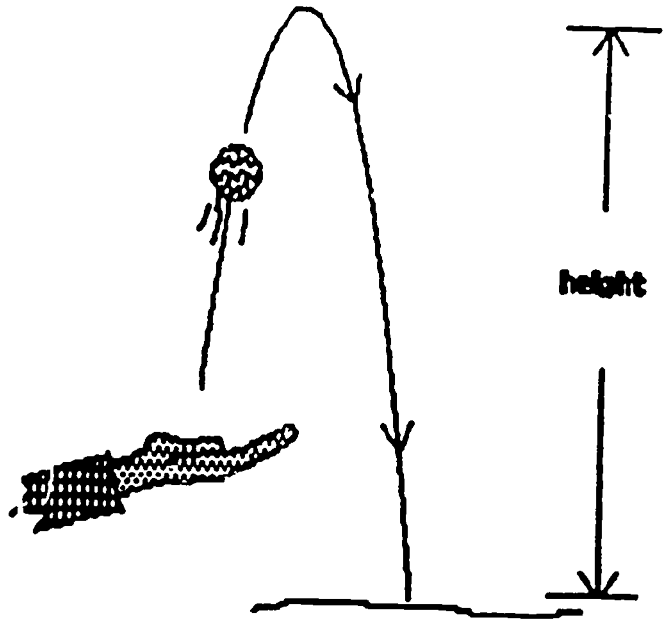

Figure 1: Figure 1: The Stone Throw Situation.

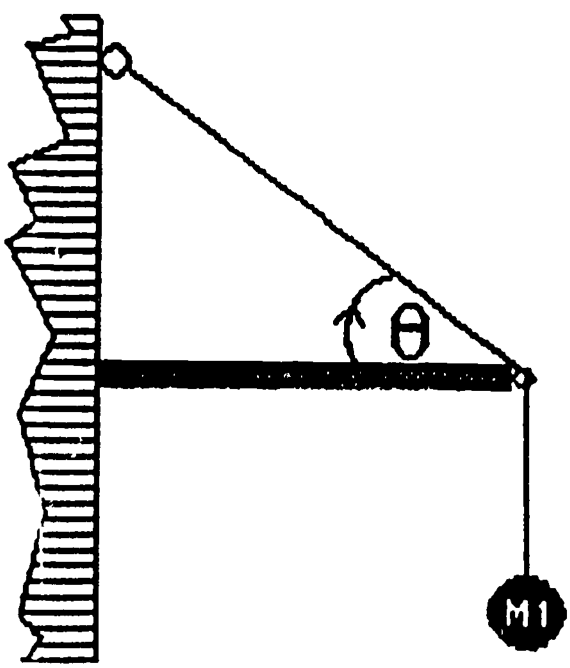

Figure 2: Figure 2: The Crane Boom Situaiton.

Qualitative Questions for the Stone Throw Problem

1. Will a lighter stone, given the same initial velocity, go higher?
2. What are the forces on the stone when it is half way up? At the top?
3. Does it come down faster than it goes up?
4. Is the time between the throw and the apex longer than between the apex and the impact?
5. Why does it reverse direction at the top?
6. Can it be thrown so hard that it will not fall back?

Quantitative Questions for the Stone Throw Problem

1. Given initial velocity, what is the maximum trajectory height?
2. If the stone is released from a height of 3 meters, what is the final velocity?
3. What initial velocity must I give it so that it will be going 30 m/sec 20 meters from the ground on the way back down?
4. If I release a second stone just as the first one passes my hand on the way down, what will be the time lag between their impacts?

Figure 3: Questions for the Stone Throw Problem.

building a computer tutor.

3. Types of Knowledge in Tutoring Physics

As already mentioned, the KR of a computer tutor will vary with the kind of reasoning and teaching the system is expected to perform. In this section we propose an orthogonal categorization to that of the quantitative and qualitative tutor. In this section, we categorize domain knowledge for a computer tutor into facts, skills, and concepts on the one hand and into expert and pedagogic knowledge on the other hand. We will also discuss knowledge representation issues in student modeling and in the next section describe KR for two theoretical tutors in terms of this taxonomy.

Qualitative Questions for the Crane Boom Problem

1. What are the forces on the cable (names and directions)?

2. Given any cable strength and bob weight, can we always find a theta which will break the cable? (The answer is yes, I think)

3. How does the weight of the boom effect the torque on the wall?

4. In what direction does the boom push or pull the cable?

5. What would happen if the cable and bob wire were one continuous wire and it ran through a metal "eye" at the end of the boom that allowed it to move freely?

6. What would happen to the tension in the cable if the boom angle were increased?
   If the boom were longer? If the bob wire were longer?

Quantitative Questions for the Crane Boom Problem

1. Given the bob weight and cable angle theta, what is the tension in the cable?

2. If the cable breaks at 500 lbs., what is the min. theta?

3. For a theta of 30 degrees, what is the maximum weight?

4. Given theta and the cable tension, what is the bob weight?

5. If the cable is shortened to raise the boom to a 45 degree angle, what will be the tension on the cable?

6. What is the force on the wall by the boom?

Figure 4: Questions for the Crane Boom Problem.

3.1 Facts, skills, and concepts

The first division of domain knowledge is into facts, skills, and concepts. Facts include declarative knowledge, including propositions, definitions, objects, properties of objects, relationships between objects, etc. Skills include knowledge about how to use factual knowledge and might include procedures and heuristics. In many domains, skill knowledge is the problem solving knowledge. Concepts include entities that act as place-holders or denotations to complicated constructs which we believe experts have, such as knowledge about gravity. Novices, too, hold concepts, however their concepts often include misconceptions. Thus we use facts, skills, and concepts as a convenient way to tease apart the knowledge of the domain. Although our taxonomy of knowledge is independent of the way it is represented in a computer program, it is worth noting that facts are often represented using frames or semantic networks and skills as a set of production rules or as attached procedures in frames systems.

Many KR languages (and cognitive science theories) propose breaking knowledge into declarative or procedural components corresponding to the first and second category above. In tutoring, however, we find it necessary to include the third category, concepts, because some knowledge, due to its complexity, can not be definitively represented. Examples include force or conservation of momentum (in physics), recursion or types (in programming), and symmetry and infinity (in math). These concepts can only be described in terms of how they relate to other pieces of knowledge. For example, the declarative knowledge of how many planets there are in our solar system can be stored explicitly in our KR, as can procedures or heuristic rules for solving simultaneous equations, but the concept of force can only be described by examples, by relating the concept to other concepts, listing related facts and procedures, and/or specifying behaviors that indicate competence with the concept.

3.2 Expert knowledge and pedagogic knowledge

Implementation of a tutor requires knowledge used by an expert to solve problems in the domain along with pedagogic knowledge about how to teach the expert knowledge. Pedagogic knowledge includes knowledge of the importance of items, the salience of features, the necessity or typicality of features, the learning difficulty of procedures, and the necessity or sufficiency of rule antecedents. Some pedagogic knowledge is associated with problem solving rules and allows an expert system to be articulate, i.e. to explain its actions and decisions. Illustrative examples and diagnostic procedures are also part of pedagogic knowledge.

Pedagogic knowledge may be represented separately from expert knowledge, or it can be sprinkled amongst the expert knowledge-- included in the frames and production rules of the expert knowledge. Relating this categorization with the previous one, fact and skill knowledge can be of either an expert or a pedagogic nature, while concepts will always be pedagogic entities, since they are not needed by an expert system to solve a problem.

In sum, the knowledge necessary for a computer expert to solve a problem is expert knowledge; all other knowledge (used for explanation, tutoring, student modeling, etc.) we call pedagogic knowledge, because it is needed in order to teach or explain the expert knowledge. Pedagogic knowledge contains more than information about "how" to teach the domain knowledge; it contains information about how the student learns or fails to learn that information. Common student errors and misconceptions are often contained in the pedagogic knowledge. Novice students will often focus on inappropriate (yet correct) features of a situation.

For example, given a book resting on a table, a student may think that a table's "inflexibility" property implies that it can not exert a force. Such non-relevant properties of objects should be included in the tutor's KR, and flagged as non-relevant. Also included as pedagogic knowledge would be such things such as misconceptions, curriculum sequences, prerequisites, "genetic" or evolutionary links between KR items (such as generalization, specialization, refinement, etc., which characterize the process of learning by modifying existing knowledge [Goldstein 1982]), and pedagogic links between concepts or examples (such as extreme case, analogy, counter example, etc.).

3.3 KR for a Student Model

Our discussion of student modeling addresses only the representation of student knowledge, and not the more difficult issues in student modeling of credit/blame assignment and student plan recognition. There seems to be two general methods of modeling the student in existing ITS systems. One method models the student knowledge in relation to the expert knowledge (we will call this "clone" modeling), and the other method does not use expert knowledge or represents student and expert knowledge independently (we will call this "autonomous" modeling).

Basically, the autonomous modeling methods specifying all of the items that the student needs to learn in a check list or table form and associates a value with each of these items. The values can be "yes," "no," or "unknown." The values can also be numerical certainty or strength factors, indicating the tutor's certainty that the student has the piece of knowledge, or alternatively, the estimated strength of the knowledge in the student. Some schemes have included records of the number of times a piece of knowledge was needed, used successfully, used unsuccessfully, and ignored. Mis-knowledge, such as repeated errors and misconceptions, can be similarly represented in autonomous student modeling. Autonomous modeling must be used in tutoring systems which are not expert systems, i.e. in systems whose knowledge objects merely refer to human knowledge, and do not attempt to represent that knowledge computationally.

Clone modeling is more complicated; the student model is linked directly with an expert knowledge representation. In such a system the student's behavior can be compared with the way the expert would have solved a problem or answered a question. The student's knowledge is interpreted as the union of: 1)a subset of the expert knowledge (an "overlay" model), and 2) a set of "buggy" items. The buggy items are perturbations (like mutant clones) of the expert knowledge. There are a number of simple perturbations types. They include (in frame terminology): adding and deleting values of features, and adding and deleting features (slots). For example, assume that the Dog frame has a slot named Legs which has a value of 4. Examples of errors are: "Dogs have 5 legs" (value of 5 in the Legs slot), "I don't know how many legs dogs have" (deleted value), "Dogs have 2 wings" (a feature Wings is added, with its value of 2), and "Number of legs has nothing to do with dogness" (a missing feature, Legs).

Other types of deviations involve hierarchical or evolutionary relationships between pieces of knowledge. Examples are overgeneralization ("Animals have 4 legs"), and inappropriate refinement (distinguishing between flexible and inflexible objects when inappropriate) or inappropriate analogies.

Some errors involve the interaction of normally unrelated pieces of knowledge. Examples are using the name of one thing to refer to another, and substituting part of what is known for an unknown item. Knowing that the student has confused two things is much more useful in tutoring than just knowing that she got a question wrong or has the wrong value for a feature.

Factual knowledge, especially if implemented in frames, is quite amenable to the deviant cloning methods, as the addition, modification, and deletion of values and slots above suggests. Skill knowledge is also amenable to cloning, as suggested in the Burton [1982]. In modeling student skills, in addition to representing how accurately the student knows the individual steps, the order of execution and the perceived priority of steps or rules is also significant.

Conceptual knowledge consists of referents to human knowledge constructs and as such can't be modified computationally. However, as suggested above, links amongst concepts and misconceptions can indicate whether they are known to be generalizations, specializations, etc., of each other.

The expert knowledge in the tutor is fixed (during a tutoring session), while the student model is constantly updated. The uncertainty involved in modeling the student puts strong demands on KR, and may require truth maintenance or endorsement information [Cohen 1985].

Finally, it may be desirable to include student background information, such as learning styles and preferences, tutoring session history, and personal and academic information.

4. KR for Tutors for Qualitative vs. Quantitative Thinking

The previous section has provided some terminology with which to discuss KR systems for tutors that focus on qualitative and quantitative reasoning. In this section, we present designs for either the qualitative or quantitiative physics tutor and strongly distinguish the two inorder to clarify our position. In actuality, the distinction between qualitative and quantitative reasoning is not always clear and, as mentioned above, the ideal tutors will have abilities in both areas.

The design for our quantitative tutor will enable it to be an expert problem solver and its KR will focus on factual and skill knowledge. The design for our qualitative physics tutor will not be able to produce an expert problem solver and its KR will emphasize conceptual as well as pedagogic knowledge.

4.1 KR for a Qualitative Physics Tutor

Qualitative understanding of a complex domain is essential to the process of learning. However, it is very difficult to build an expert qualitative problem solver. DeKleer & Brown [1980], and Forbus [1982], and others have researched modeling qualitative processes (and "mental models") and have looked in detail at how to answer qualitative questions about such objects as car engines, electrical circuits, projectile motion, and fluid circuits. They are interested in questions such as "How does it work," "If I increase this what will happen to that," "What causes this part to function," and "What will happen if I take this part out?." This work looks promising, but it is still in its formative stages. As their research progresses, we may eventually be able to incorporate expert qualitative problem solvers in physics tutors. Our design for a qualitative physics tutor does not result in an expert system. In Figure 5 we provide example frames using our proposed qualitative-KR structure. We store pre-calculated solutions and text explanations, and focus on the pedagogic knowledge needed to: 1) convey the concepts to be taught, and 2) recognize and respond appropriately to preconceptions and misconceptions.

The qualitative-KR scheme we present here comes from the case based tutoring (CBT) paradigm which we are developing. CBT is an example based Socratic tutoring style that emphasizes qualitative and analogical reasoning and appears effective for tutoring subjects where student's preconceptions play a large role. A first- order case based tutor will run without language recognition, language generation, sophisticated discourse knowledge, example generation, sophisticated student modeling, or an expert problem solver. Therefore much of the information in the qualitative-KR is in the form of canned text and pre-defined pointers. More advanced systems need to have domain inferencing and communication knowledge which will enable the tutor to do some of these things intelligently.

CBT focuses on the use of example situations in tutoring. The types of objects in the qualitative-KR system are Situations,*Sit-rels* (situation relationships), *Concepts*, and *Misconceptions*. Situations contain descriptions of the example situations, questions and answers about these situations, and information about the situations relevant to using them in tutoring, such as prerequisite concepts, key assumptions, and level of difficulty.

The *Sit-rels* describe how the situations in the qualitative-KR are related pedagogicaly. Examples are extreme case, simple case, analogous case. The Sit-rels are used by (yet to be specified) tutoring rules in deciding what situation to present next.

Concepts and *Misconceptions* are objects that form part of the student model. A student's answer to a question may increase the evidence that a student has or does not have a certain concept or misconception. Concepts and Misconceptions merely refer to hypothetical constructs in the minds of students or experts, they do not explicitly describe these constructs. It may also be possible to have objects called Concept-rels which relate concepts and misconceptions according to relationships such as prerequisite, generalization, specialization, over-generalization, etc. [Goldstein 1982]. It is not clear how this type of knowledge should be used though.

Pedagogic knowledge at the curriculum level could be included to structure the presentation of material. The tutor could follow this curriculum, deviating from it when remedial or explanatory action is needed (which should be quite often), or when the student interrupts the tutor with, for example, a request for a new example. The following object types are possibilities:

Topics - Define the local goals of the tutor by organizing the concepts to be addressed with key example situations. Topics also have information about prerequisite concepts and topics, pre-tests, summary and overviews, and post-testing.

Curricula - A partially ordered set of topics.

(situation-1 vertical-stone-throw

(key-concepts (force-existence 1-dim-projectile-motion))

(description "a person throws a stone straight up in the air, and it eventually lands next to them")

(detailed-description "a stone is thrown almost straight up. It reaches its peak
and then comes back down landing next to the thrower. We assume there is
no effect from air friction.")

(assumptions ("no air friction"))

(question-1 "what are the forces on the stone when it is half way up?"

(explanation "The downward force of gravity is the only force on the stone while it is airborne. ")

(hint-1 "remember that a force is any kind of a push or a pull")

(correct-answer #1)
(answers
(a1 "there is only the constant force of gravity acting down"
(miscon nil) (concept 3))
(a2 "there is only the downward force of gravity which is always increasing"
(miscon 16) (concept 3))
(a3 "there is the downward force of gravity and the upward force of the throw
which is constant"
(miscon 1) (concept nil))
(a4 "there is the downward force of gravity and the upward force of the throw
which is decreasing."
(miscon (1 2)) (concept nil)))
(misconception-1 "objects carry an impetus force")

(misconception-2 "impetus force dies out with time")

(concept-3 "objects don't carry an impetus force")

(sit-rel-1

(simple case (vertical-stone-throw stone-drop))
(expl "only gravity is acting in both cases"))

(sit-rel-2
(extreme-case (vertical-stone-throw throw-to-moon))
(expl "if there were no other forces on the stone, it would always come
back because there is always some force of gravity on it. However,
other forces, like the moon's gravity, could change its trajectory "))
(sit-rel-3
(comparison (vertical-stone-throw rocket-ship-liftoff))
(expl "a rocket ship liftoff is different because the engines
provide a constant force on the rocket"))

Figure 5: Example Qualitative Knowledge for the Stone Throw Situation.

" 12" was mistakenly skipped in page numbering.

4.2 KR for a Quantitative Physics Tutor

Our goal is to design a quantitative-KR structure that will be general and powerful enough to support (or upgrade with) the inferences made by tutors and expert problem solvers (within tutors) for many domains. The structure we suggest may be an over-kill for any particular tutoring domain and it remains to be seen if the extra effort is worth effort of trying to achieve uniformity across many tutors.

The quantitative-KR system we propose is based on using a frame-like shell with nominal functionality including inheritance, defaults, and attached procedures. In addition, we keep within the spirit of KL-ONE [Brachman & Schmolze 1985] knowledge representation framework, and define a system that limits the set of possible relationships or slot names for objects. Doing so provides greater precision in representation and increases the power of the inference rules and operators which act on the knowledge. According to this model, we limit the slot names of objects to things like is-a, subparts, properties (1-place predicates), relationships (n- place predicates), and examples.

For example, in the non-ideal case, if we want to represent that a table is blue, we could have a slot for Color and fill it with "blue" for this particular table. A more desirable representation would be to have an object called Color which is of type Object-property, and have an instantiation of Color, called Color-15, which represents the property of having a blue color. We then have a slot in the table object for object-properties. One of the items in this slot would be Color-15. This extra level of refinement enables us to reason about colorness in general through the Color object, and to reason about blueness in general through the Color-15 object. Other physical objects that have a blue color would have this same object, Color-15, in their object-properties slot.

4.2.1 Representing Facts

Our ideas for a quantitative-KR structure for a physics tutor were inspired in part by the works of Forbus [1982], deKleer & Brown [1980], and Novak [1977]. We represent separately noun-like objects, such as physical objects and states, from relationship and property objects, so that an expert system can reason about properties and relationships. Figure 6 define classes of objects for our proposed quantitative knowledge base and Figure 7 provides example frames from the stone throw problem.

4.2.2 Representing Skills

The tutor needs procedural knowledge (skill knowledge) about how to solve problems in the domain. This knowledge can range from simple reasoning about specific domain algorithms or equation solving, to more complex heuristic problem solying knowledge or spatial/geometric reasoning, to very complex reasoning about "common sense" knowledge. Common sense knowledge, such as "if A is on top of B then A touches B" and "if A is red then A can not be blue," is in general, difficult to model with completeness. However, some common sense knowledge can be easily incorporated into frame representations. For example, all baseballs are kinds of balls (inheritance), all balls are round (slot values), carts

Phys-objs (physical objects) represent the physical objects in the problem such as cars, people, balls, planets.

Can-objs (canonical objects) represent abstractions of physical objects that have significance in physics problem solving. All Phys-objs are of some type of Can-obj. Examples of can-objs are: cart, ball, inclined plane, pivot-point, point-mass, wall. For example, cars, trains, wagons, etc. are all treated the same for the purpose of physics problem solving, so they are all or the same type, i.e. "cart."

Obj-props (object properties) represent physical and canonical objects defined according to their properties by pointing to obj-props. The obj-props themselves are separate objects which contain information about things which must be true of any object having that property. Can-objs have physics-relevant properties such as weight, length, surface-smoothness. Phys-objs have non-relevant properties such as color, and owner.

Obj-rels (object relationships) represent specific relationships between objects. Examples are on-top-of, touches, distance-between, attached-to.

Formula-rels (formula relationships) represent relationships between parameters of objects. An example is "Newton's-second-law," F=ma. One could specify that the Newtons-second-law relationship holds between force-on(ball), mass-of(ball), and acceleration-of(ball). Here the force, mass, and acceleration are Obj- props of the Can-obj ball.

P/S's (process/states) represent physical processes or states such as rolling, sliding, evaporation, oscillation, collision, and electric current. P/S's have slots for "actors" which are filled by Phys-objs. P/S's specify quasi-stable relationships between objects by indicating what Obj-rels hold between the objects in that P/S, and what Formula-rels hold for the parameters of the objects. The P/S can also specify functional, causal, and temporal relationships between objects. The crane-boom is a P/S since it has a given set of relationships between a given set of objects (even though the parameters can change their values)

Situations are physical situations composed of one or a number of P/S's which share common actors and have an overall temporal or causal connection. Situations are included in our scheme so that we can represent entire stories or a sequence of P/S's. For example, a block slides down an inclined plane, falls, and lands on a sponge. The sliding, falling, and collision are separate P/S's. Heating, evaporation, convection, cooling, and then precipitation of water is another example of a Situation comprised of several P/S's. The individual P/S's share some of their objects (actors) in these Situations. For most of the examples we are concerned with, the Situation consists of only one P/S, so for most of this paper the words Situation and P/S can be used interchangeably, i.e. a reference to the crane-boom situation is equivalent (unless otherwise noted) to the crane boom P/S.

Figure 6: Representing Facts in a Quantitative Knowledge base.

```
(phys-obj-stone-1
 (canonical-type point-mass)
 (properties
  (mass
   (value 50) (units kg))
  (material rock)
  (color gray)))

(phys-obj-person-1
 (canonical-type cannon)
 (properties
  (name "a person") (angle 90)
  (humanp t)))

(can-obj-point-mass
 (properties
  (mass nil)))  ; nil for unknown

(can-obj-cannon
 (properties
  (angle nil) (max-force nil)))

(formula-distance
 (parameters (x a t vo xo))
 (relationship "x = .5 * a * t*t + vo * t + xo"))

(formula-velocity
 (parameters (v a t vo))
 (relationship "v = a * t + vo))

(formula-newtons-3rd
 (parameters (f m a))
 (relationship "f = m * a"))

(formula-gravity
 (parameters (m1 m2 g r f))
 (relationship "f = g * m1 * m2 / r*r"))

(p/s-vertical-projectile
 (name nil)
 (actors (cannon point-mass))
 (parameters
  (height nil)  (vel nil)  (init-vel nil)
  (mass get-prop point-mass mass) (cannon-angle get-prop cannon angle)
  (cannon-height nil)  (a (value -5) (units m-per-sec-squ))
  (final-vel nil)  (apogee-height nil))
 (relationships
  ((formula-distance (height a t init-vel cannon-height)
  formula-velocity (vel a t init-vel)
  (setq apogee-height ((-vo*vo/(2*a) + xo ))))
```

Figure 7: Example Quantitative Knowledge Representation for the Stone Throw Situation.

usually have four wheels (defaults), and the center of a board is half its length from one end (attached procedures).

An ideal computer tutor needs procedures and/or rules that enable it to make sophisticated inferences in order to 1) understand, 2) solve, and 3) teach domain knowledge and domain problems. As mentioned above, we are not concerned with tutoring rules (item 3) here (except to note that pedagogic knowledge about the expert knowledge will assist in student modeling and will help dynamically determine the foci and goals of the tutoring session). We will also assume that the tutor understands the meaning of the problems that it generates or is given (item 2; i.e. it does not have to interpret the meaning of an arbitrary domain problem given to it by the user). So we will limit our discussion of problem solving knowledge to those skills needed to solved the problem once it is understood.

Several researchers have looked at the procedural nature of skills as represented in tutors [Burton 1982, Goldstein 1982, and Anderson 1985]. In such systems, student behavior is usually modeled as if-then rules (productions), and the student's skill knowledge is interpreted as a set of rules that overlap the expert rules. Some student actions are interpreted as correct expert rules, and others as erroneous rules. The wrong rules can be deviations of expert rules or extraneous rules. The tutor follows the student's behavior, interpreting it in terms of recognizable correct and "buggy" rules, and then takes appropriate action. As Anderson points out, problem solving activity is goal-driven. The productions in his tutoring systems reflect this by indicating how goals are broken into subgoals as the problem's "solution space" is searched. He stresses the importance of communicating the goal structure of the problem solving to the student. (See also Heller & Reif 1984 on goal structured physics tutoring).

As an example, the goal of solving a crane boom problem could be broken up into these subgoals (or sub-procedures, or sub-skills):

1. Recognize the important objects and their parts (booms, cables, etc.).
2. Recognize the relevant physical relationships between objects (connections, supports, etc.).
3. Recognize and label the important parameters of the problem (forces, lengths, etc.).
4. Decide on a strategy. For example use the fact that the sum of the forces and torques on each object is zero in a static situation.
5. Recall the relevant formulas.
6. Instantiate the formulas for this problem.
7. Solve the equations.

8. Interpret the answer in terms of the original problem situation.

These subgoals would in turn be broken up into sequences (or hierarchies) of goals or specific actions. There are many design issues about how to incorporate production rules for problem solving in tutoring systems, including making a choice of grain size for the rules, deciding on conflict resolutions strategies, and recognizing student plans and goals.

The quantitative-KR for an expert tutor may need procedural expertise in measurement (measurement error, units conversion, rate of change, etc.), geometry/trigonometry (spatial reasoning), temporal reasoning, and causal reasoning. More sophisticated systems may need to reason about hypothetical situations, possibility, and the probability, consistency, relevance, and redundancy of information.

4.2.3 Representing Pedagogic Knowledge in a Quantitative Tutor

The KR for design of a quantitative tutor contains only pedagogic information, such as concepts and examples. There are some types of pedagogic information which apply only to the qualitative tutor. For example, we may want to store pedagogic information about the importance and salience of object's properties. The non-relevant (to physics) Obj-props mentioned above are pedagogic knowledge. The rules or procedures that comprise skill or heuristic knowledge should be annotated according to difficulty level and should be annotated so that they can "explain themselves" (as in the Guidon tutor for the Mycin expert system).

Our suggestions have emphasized the use of expert knowledge for quantitative tutors and pedagogic knowledge for qualitative tutors because this division corresponds to two realizable types of tutors. As mentioned above, tutors of either type could incorporate both expert and pedagogic knowledge, and ideal tutoring systems should reason both qualitatively and quantitatively.

5. Conclusion

We have designed a knowledge representation that allows a variety of researchers to tease apart knowledge needed to teach a science domain. We have designed a structure and vocabulary to facilitate teachers and knowledge engineers in designing data bases for tutors. Though we still need to test the efficacy of our design, the intent was to amke the design 1) general enough to be used for multiple science domains and several target behaviors and 2) powerful enough to support (or upgrade to) sophisticated inferencing in expert problem solving and student diagnosis.

We introduced two independent categorizations of knowledge: facts, skills, and concepts, and expert/pedagogic knowledge. These orthogonal categories have been useful for discussing the function of pieces of knowledge, the form in which such knowledge will be implemented in an AI system, and the target behaviors desired in the students.

We have also made a distinction between qualitative and quantitative thinking in physics, and proposed KR schemes for both. This division corresponds to two distinguishable (though not exclusive) pedagogic styles and highlights the usefulness of the terminology and representational structures that we are developing. A more general KR scheme would be a straightforward combination of the features of the qualitative and quantitative schemes.

An intelligent tutor must handle large amounts of domain information if it is teach in a robust and powerful manner. The system must be managable by tutoring rules and be modifiable by teachers or domain experts. We can not expect our prototype systems to scale up to the ideal robust without considerable research. This paper shows a first step at such research and we look forward to its evolution as we attempt to utilize it for building tutors in many domains.

References:

Anderson, J., Boyle, F., Yost, G. (1985). "The Geometry Tutor". IJCAI Proceedings, 1985.

Brown, J. S., & Burton, J, (1982). "An Investigation of Computer Coaching for Informal Learning Activities." Intelligent Tutoring Systems, Ed. by Sleeman, D., & Brown, J. S. Academic Press, 1985.

Brachman, R., & Schmolze, J. (1985). "An Overview of the KL-ONE Knowledge Representation System". Cognitive Science, No. 9, 1985.

Burton, R. R. (1982). "Diagnosing Bugs in Simple Procedural Skills". Sleeman & Brown (Eds.) Intelligent Tutoring Systems. Academic Press, 1982.

Cohen, P. (1985). "Numeric and Symbolic Reasoning About Uncertainty in Expert Systems". UMASS COINS technical report 85-25.

Clancey, W. (1985). "Heuristic Classification," in Articicial Intelligence, Jan. 1985.

deKleer J., & Brown, J. S. (1982). "Assumptions and Ambiguities in Mechanistic Mental Models," Xerox Palo Alto, March 1982.

deKleer, J., & Brown, J. S. (1980). "Mental Models of Physical Mechanisms," Xerox Palo Alto report CIS-3, 12/80.

Forbus, K. (1982). "Qualitative Process Theory," MIT AI memo 664.

Goldstein, I (1982). "." Sleeman & Brown (Eds.) Intelligent Tutoring Systems. Academic Press, 1982.

Hayes, P. (1985). "The Second Naive Physics Manifesto," in Brachman & Levesque (Ed.) Readings in Knowledge Representation, Morgan Kaufman, Los Altos, 1985.

Heller J, & Reif, F. (1984). "Prescribing Effective Human Problem-Solving Processes: Problem Description in Physics." Cognition & Instruction, 1984 1(2).

Murray, T. (1986). "A Case Based Tutoring Paradigm for an Intelligent Physics Tutor." Umass COINS working paper.

Novak, G. (1977). "Representations of Knowledge in a Program for Solving Physics Problems," IJCAI proceedings 1977.

Reiser, B., Anderson, J., & Farell, R. (1985). "Dynamic Student Modeling in an Intelligent Tutor for Lisp Programming". IJCAI Proceedings, 1985.

Sleeman, D., & Brown, J. S. (Ed. 1985). Intelligent Tutoring Systems. Academic Press, 1985.

Woolf, B. (1984). "Context Dependent Planning in a Machine Tutor," UMASS COINS technical report 84-21.